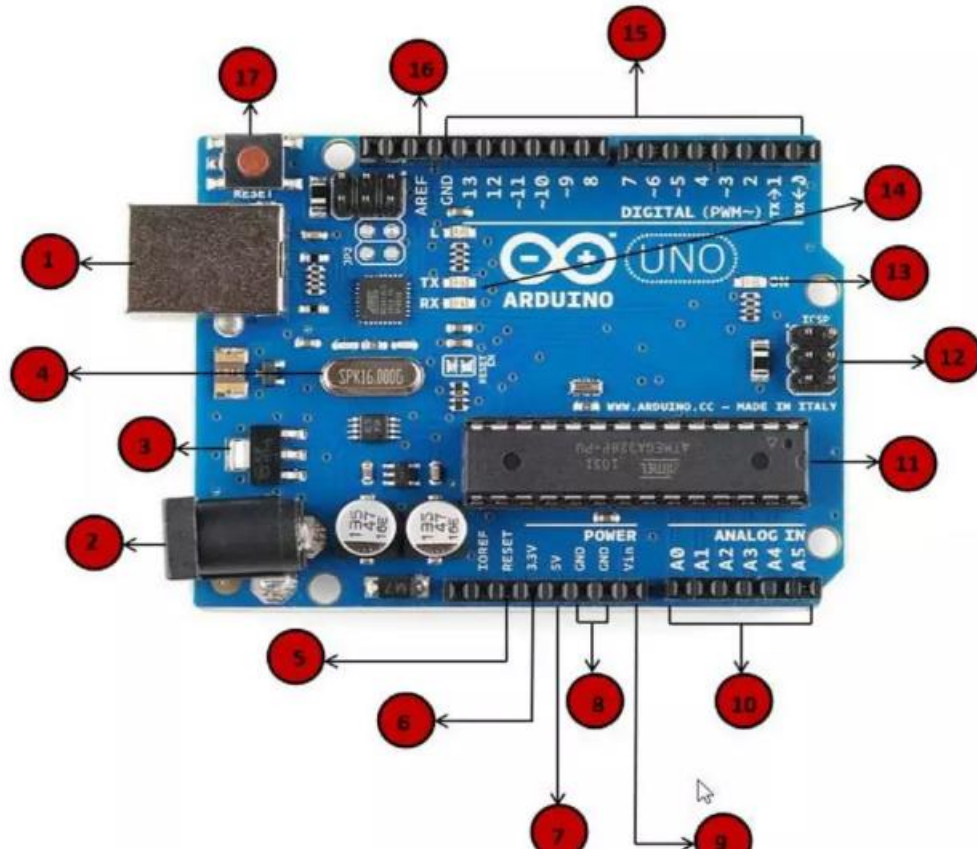


- Basic ARM architecture, ARM organization core Data flow Model,
- ARM register organization, current program register organization.
- Pin configuration and architecture Arduino,
- Introduction to Raspberry Pi, Understanding SoC architecture and SoCs used in
- Raspberry Pi, Pin Description of Raspberry Pi,
- On-board components of Rpi
- Microcontroller Applications: Interfacing matrix keyboard and
- Seven segments LED display,
- LCD Interfacing, ADC Interfacing, DC motor interfacing.

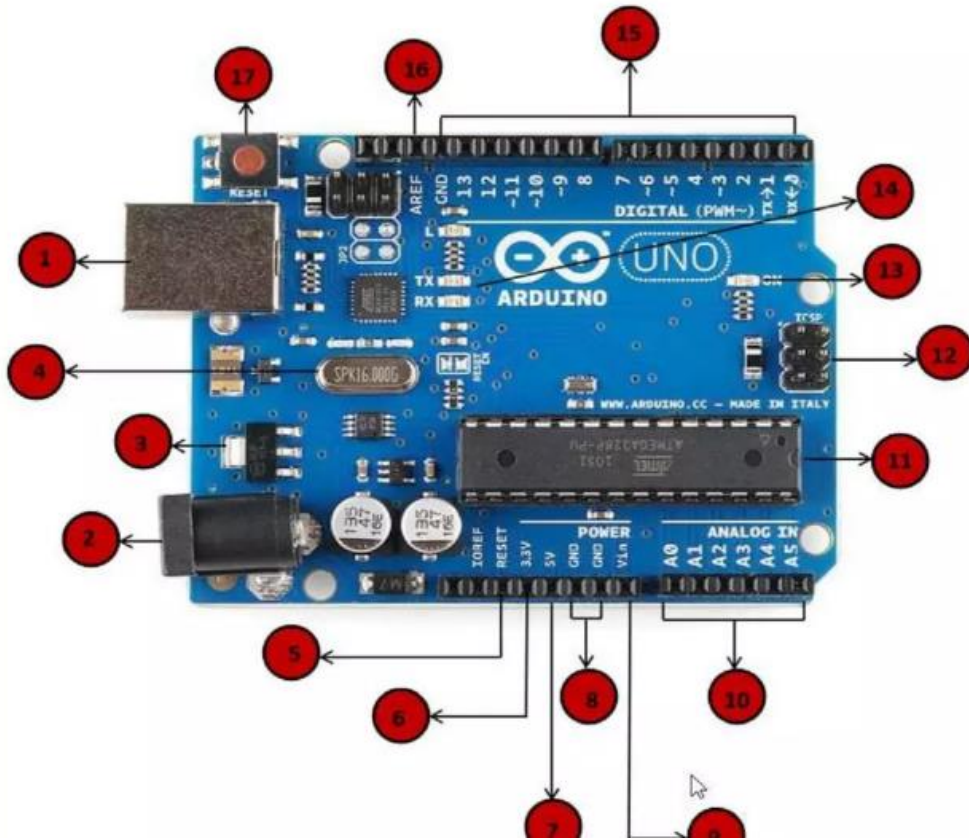
- Arduino Uno is a microcontroller board based on 8-bit ATmega328P microcontroller.
- Along with ATmega328P, it consists other components such as crystal oscillator, serial communication, voltage regulator, etc. to support the microcontroller.
- Arduino Uno has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button.

1 Power USB



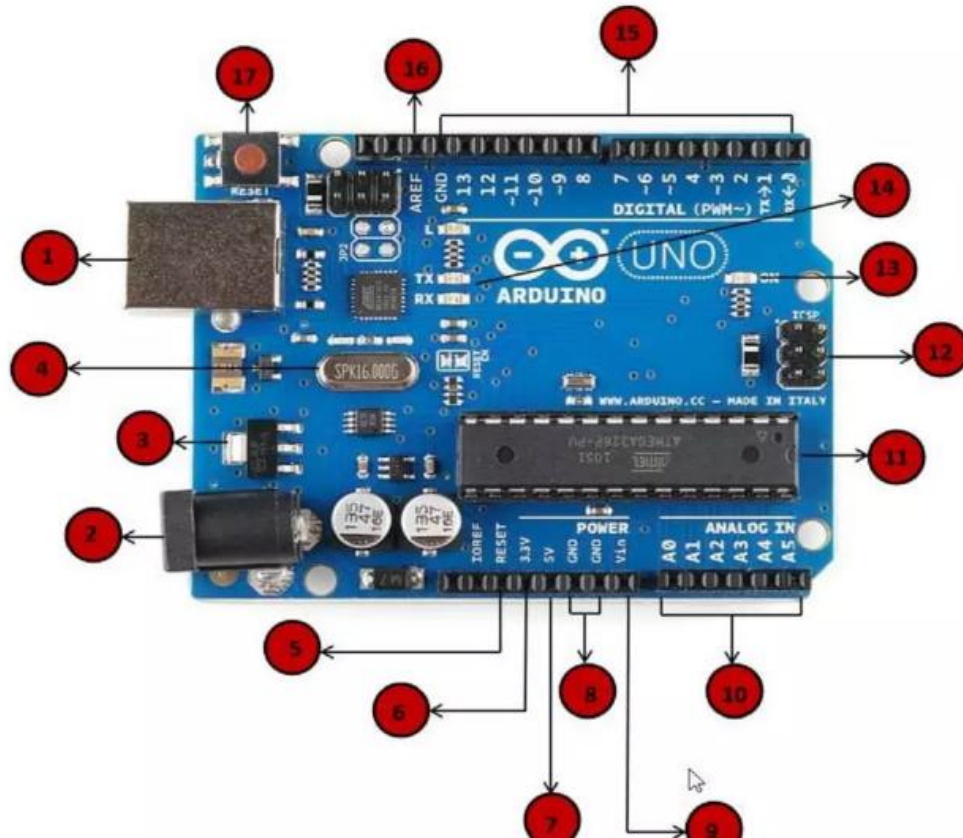
ARDUINO BOARD CAN BE POWERED BY USING THE USB CABLE FROM YOUR COMPUTER. ALL YOU NEED TO DO IS CONNECT THE USB CABLE TO THE USB CONNECTION (1).

2 Power (Barrel Jack)



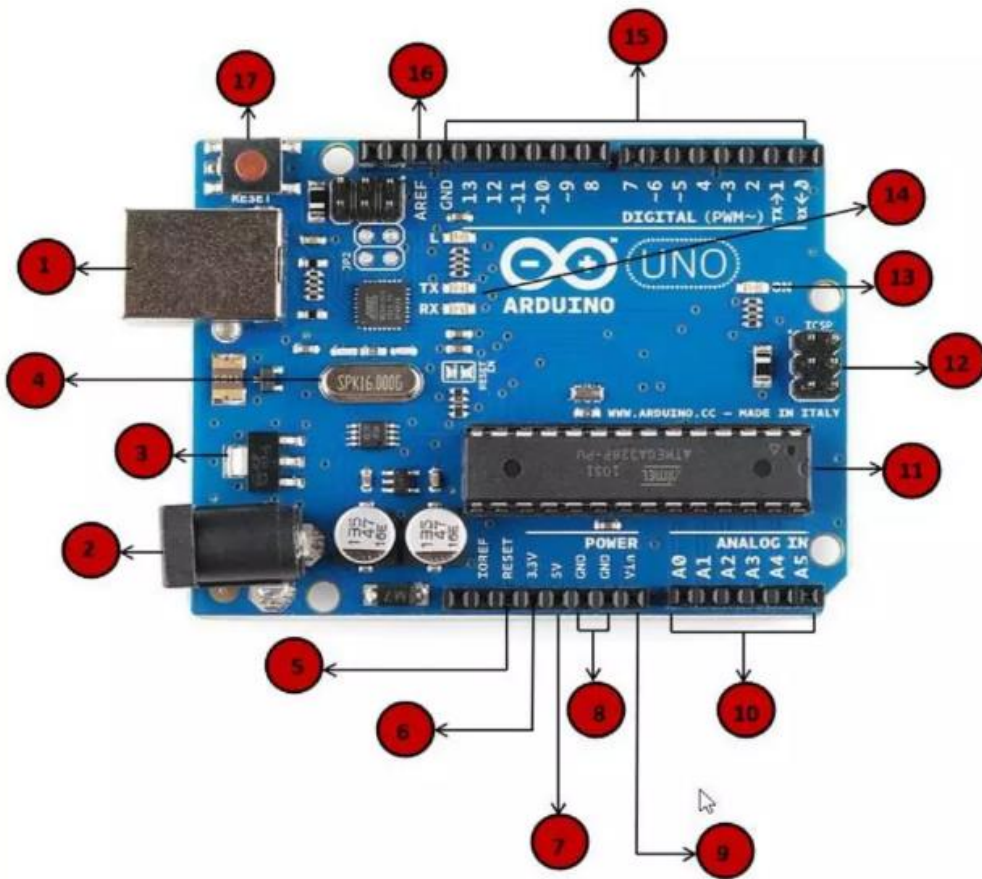
Arduino boards can be powered directly from the AC mains power supply by connecting it to the Barrel Jack (2).

3 Voltage Regulator



The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.

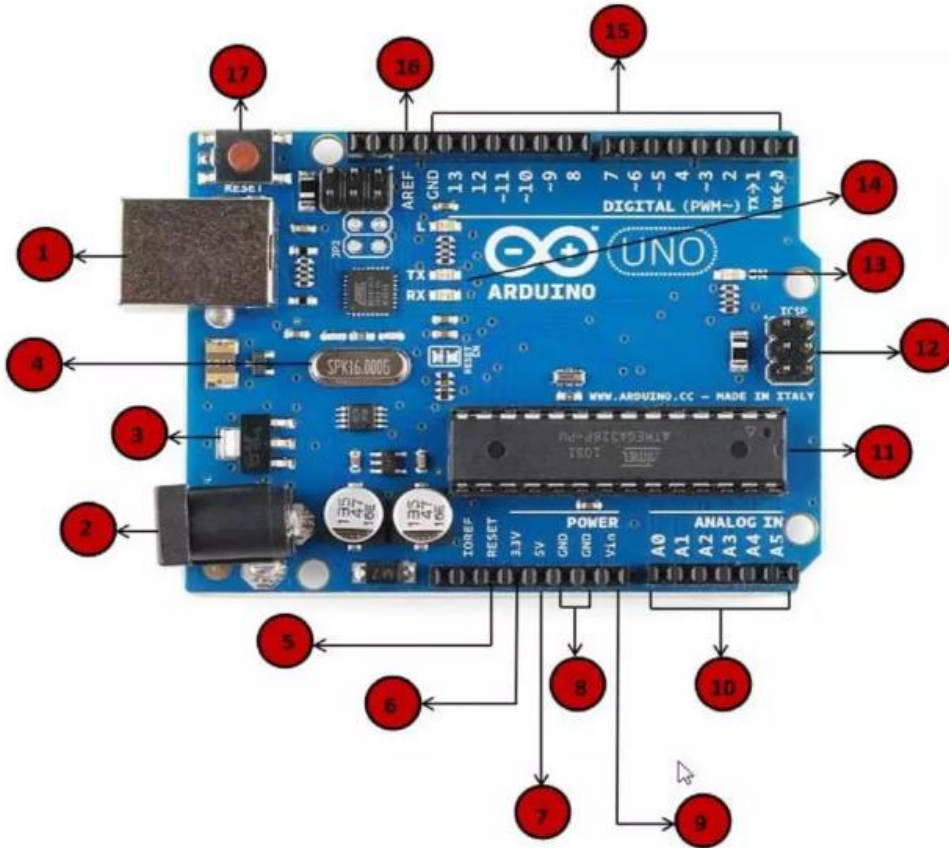
4 Crystal Oscillator



The crystal oscillator helps Arduino in dealing with time issues. How does Arduino calculate time? The answer is, by using the crystal oscillator. The number printed on top of the Arduino crystal is 16.000H9H. It tells us that the frequency is 16,000,000 Hertz or 16 MHz.

5.17 Arduino Reset

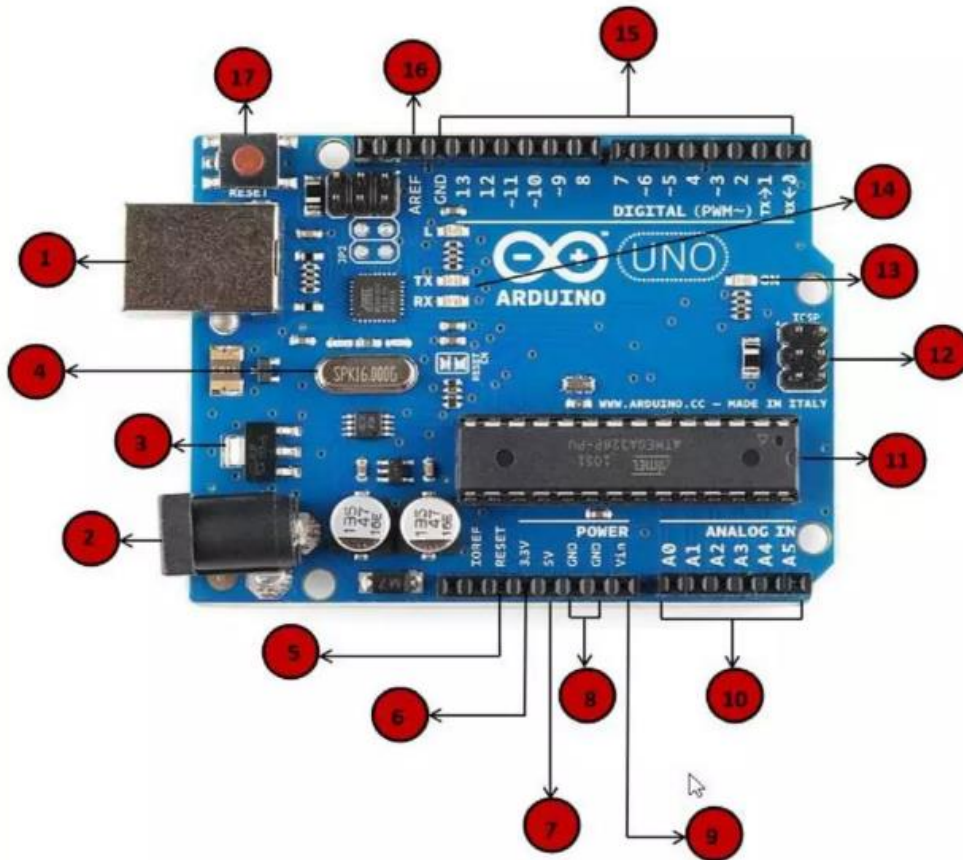
You can reset your Arduino board, i.e., start your program from the beginning. You can reset the UNO board in two ways. First, by using the reset button (17) on the board. Second, you can connect an external reset button to the Arduino pin labelled RESET (5).



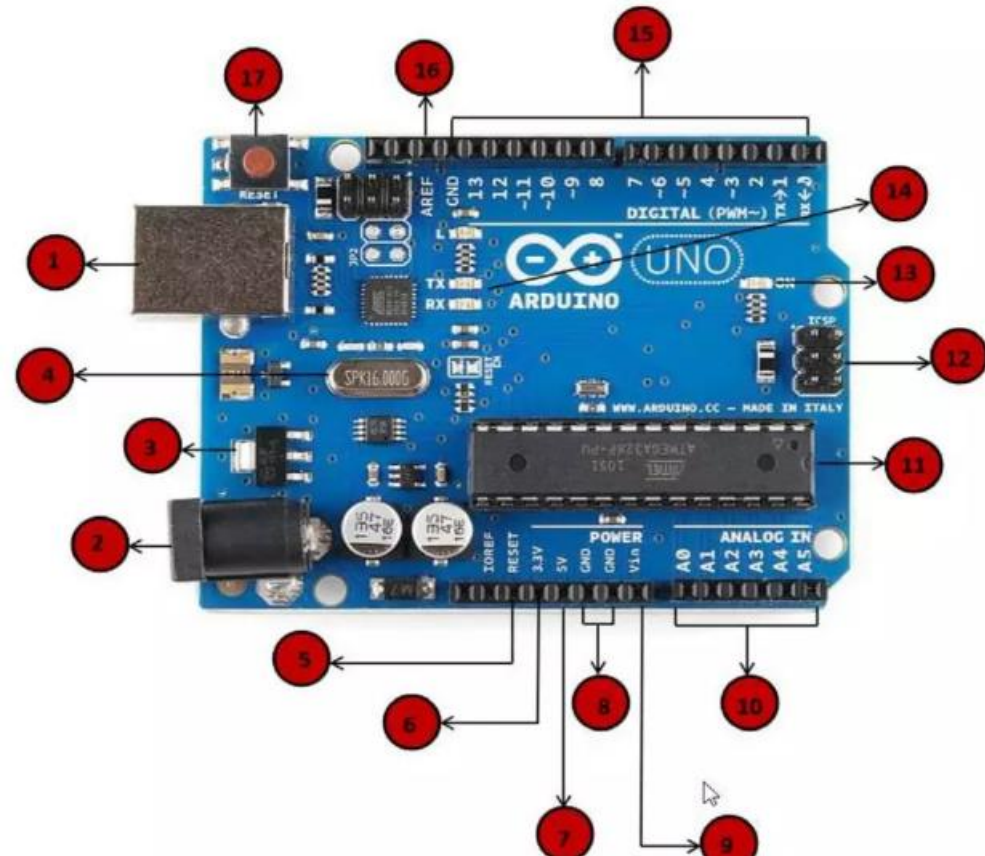


Pins (3.3, 5, GND, V_{in})

- 3.3V (6): Supply 3.3 output volt
- 5V (7): Supply 5 output volt
- Most of the components used with Arduino board works fine with 3.3 volt and 5 volt.
- GND (8)(Ground): There are several GND pins on the Arduino, any of which can be used to ground your circuit.
- V_{in} (9): This pin also can be used to power the Arduino board from an external power source, like AC mains power supply.



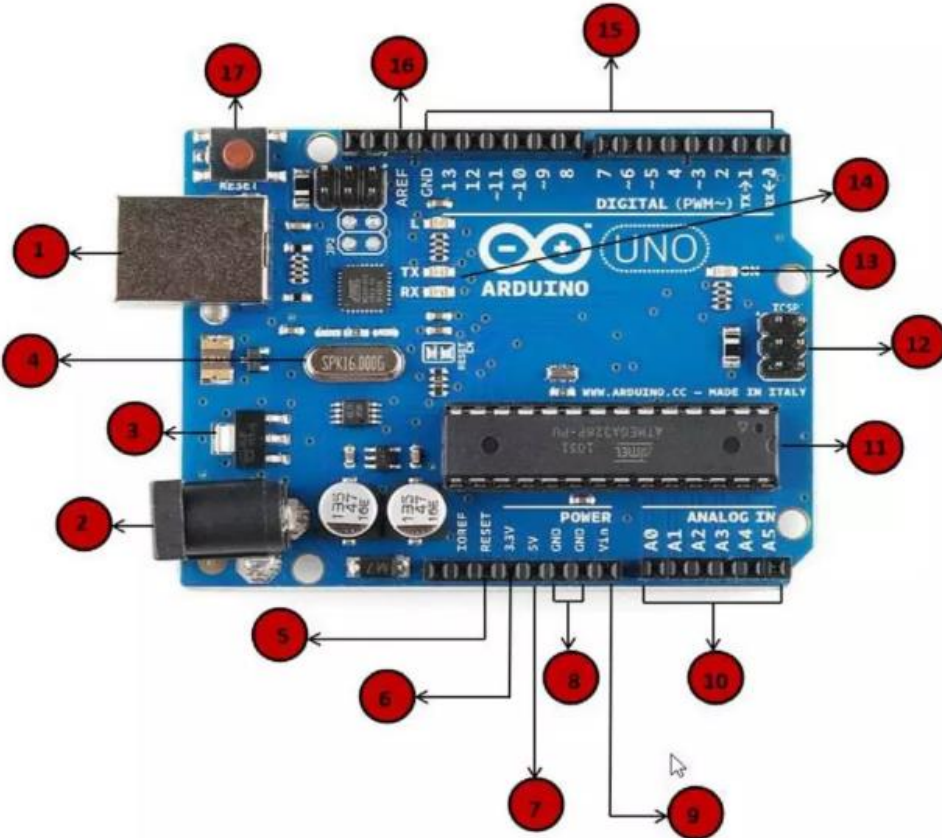
10 Analog pins



The Arduino UNO board has five analog input pins A0 through A5. These pins can read the signal from an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value that can be read by the microprocessor.

11 Main microcontroller

Each Arduino board has its own microcontroller (11). You can assume it as the brain of your board. The main IC (integrated circuit) on the Arduino is slightly different from board to board. The microcontrollers are usually of the ATMELE Company. You must know what IC your board has before loading up a new program from the Arduino IDE. This information is available on the top of the IC. For more details about the IC construction and functions, you can refer to the data sheet.



12 ICSP pin

Mostly, ICSP (12) is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It is often referred to as an SPI (Serial Peripheral Interface), which could be considered as an "expansion" of the output. Actually, you are slaving the output device to the master of the SPI bus.

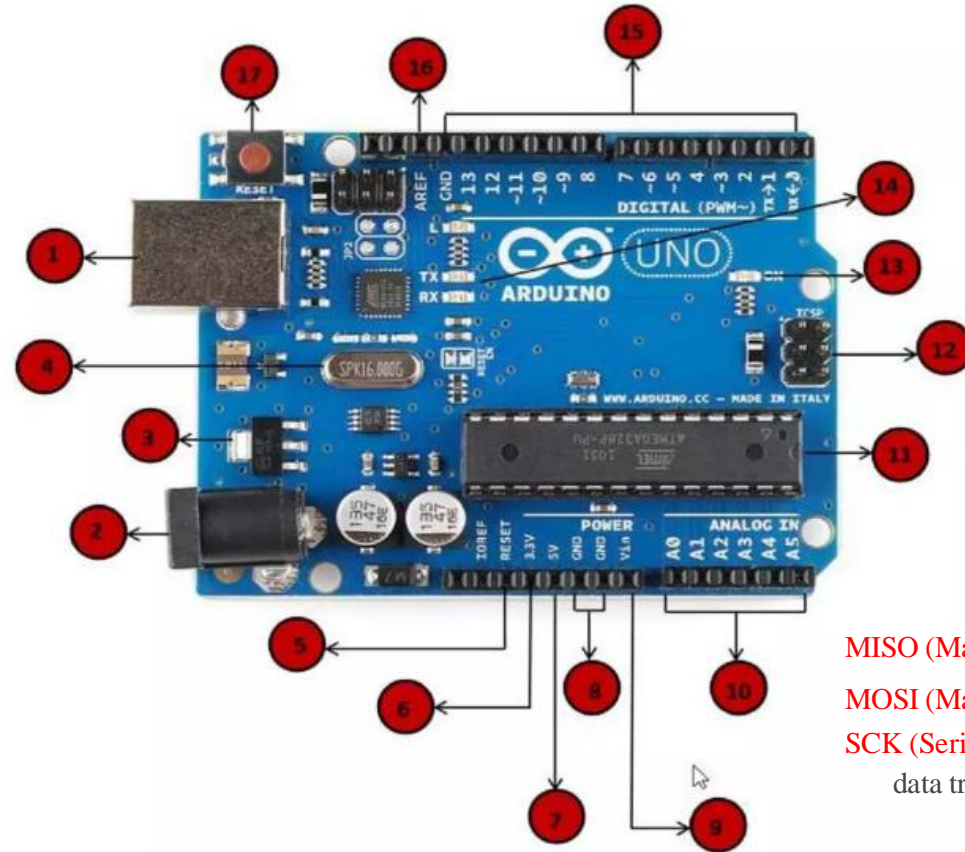
ICSP: In-Circuit Serial Programming.

AVR is a microcontroller

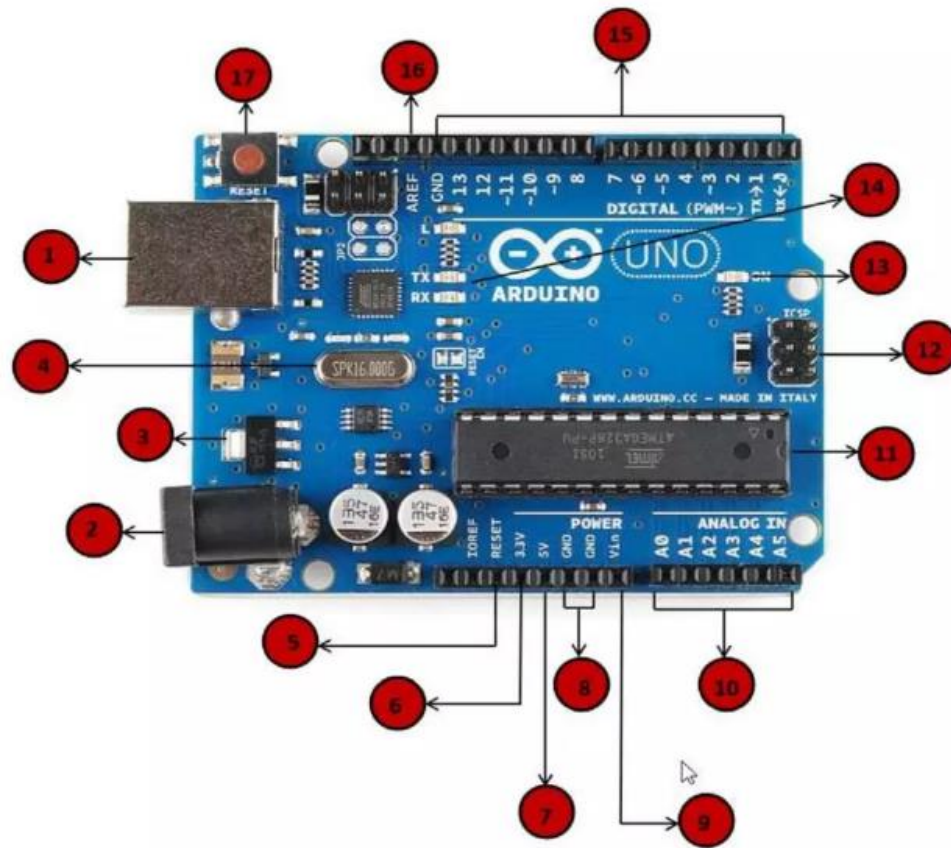
MISO (Master In Slave Out) - A line for sending data to the Master device

MOSI (Master Out Slave In) - The Master line for sending data to peripheral devices

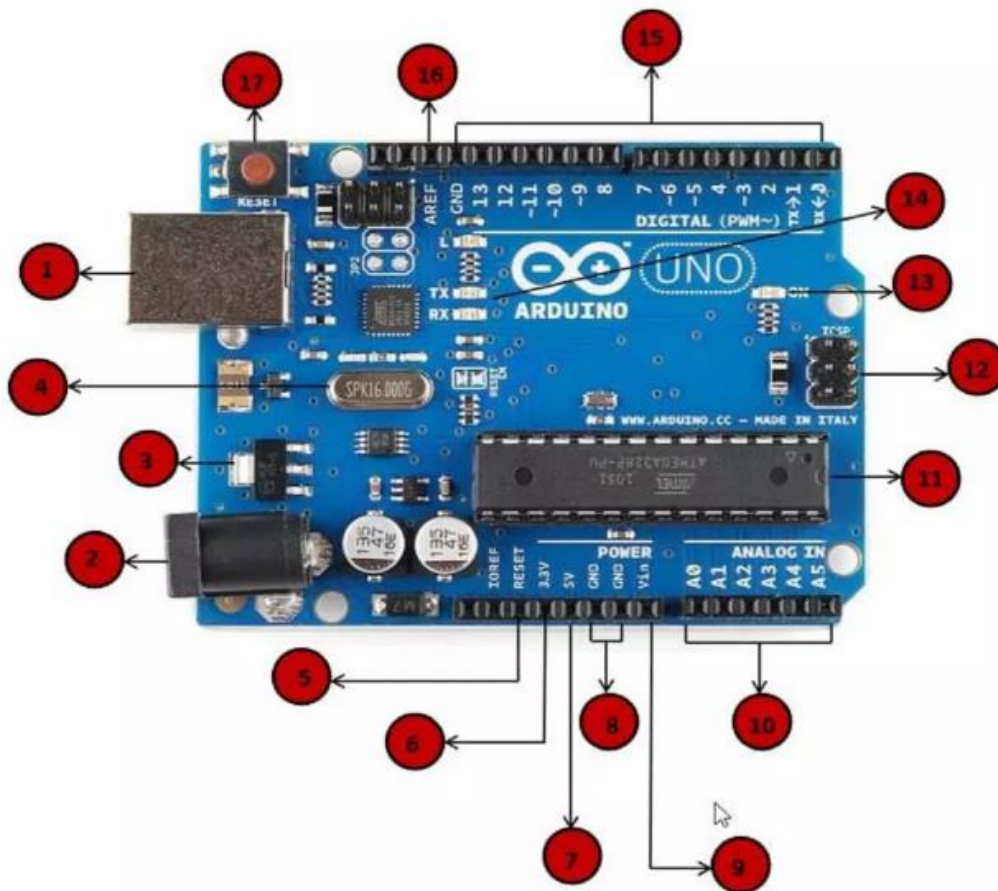
SCK (Serial Clock) - A clock signal generated by the Master device to synchronize data transmission.



13 Power LED indicator



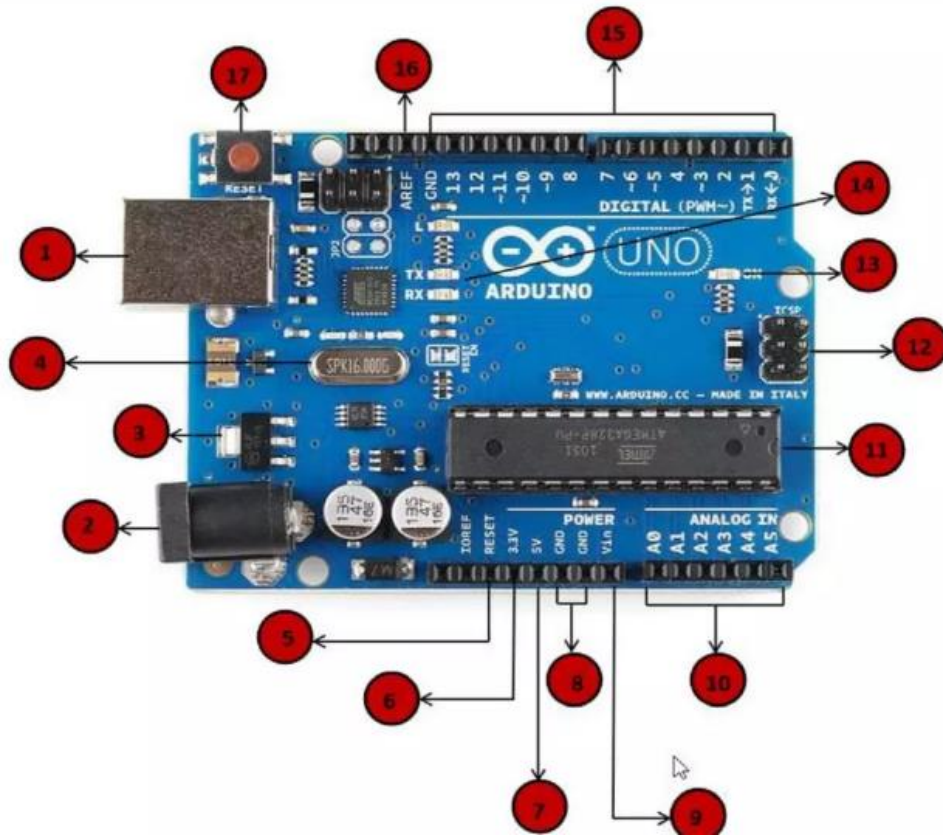
This LED should light up when you plug your Arduino into a power source to indicate that your board is powered up correctly. If this light does not turn on, then there is something wrong with the connection.



On your board, you will find two labels: TX (transmit) and RX (receive). They appear in two places on the Arduino UNO board. First, at the digital pins 0 and 1, to indicate the pins responsible for serial communication.

Second, the TX and RX led (13). The TX led flashes with different speed while sending the serial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process.

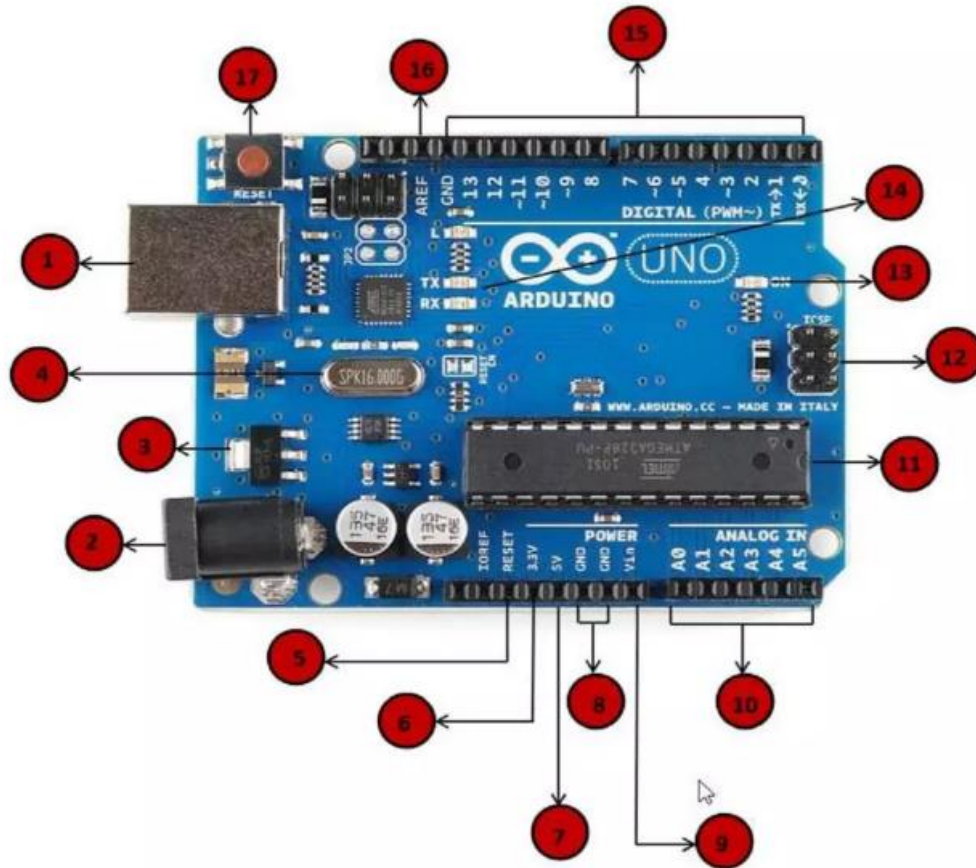
15 Digital I / O



The Arduino UNO board has 14 digital I/O pins (15) (of which 6 provide PWM (Pulse Width Modulation) output). These pins can be configured to work as input digital pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc. The pins labeled “~” can be used to generate PWM.

Pulse width modulation (PWM) is a powerful digital technique for controlling analog circuits with a microprocessor's digital outputs.

16 AREF



AREF stands for **Analog Reference**. It is sometimes, used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

The AREF (Analog Reference) pin can be used to provide an external reference voltage for the analog-to-digital conversion of inputs to the analog pins.

ARDUINO – INSTALLATION

Step 1: First you must have your Arduino board (you can choose your favourite board) and a USB cable. In case you use **Arduino UNO**, Arduino Duemilanove, Nano, Arduino Mega 2560, or Diecimila, you will need a standard USB cable (A plug to B plug), the kind you would connect to a USB printer as shown in the following image.



Step 2: Download Arduino IDE Software

You can get different versions of Arduino IDE from the Download page on the [Arduino Official website](#). You must select your software, which is [compatible with your operating system](#) (Windows, IOS, or Linux). After your file download is complete, [unzip the file](#)



Arduino IDE 1.8.19

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Refer to the [Getting Started](#) page for Installation instructions.

SOURCE CODE

Active development of the Arduino software is [hosted by GitHub](#). See the instructions for [building the code](#). Latest release source code archives are available [here](#). The archives are PGP-signed so they can be verified using [this](#) gpg key.

DOWNLOAD OPTIONS

Windows Win 7 and newer

Windows ZIP file

Windows app Win 8.1 or 10 [Get](#) 

Linux 32 bits

Linux 64 bits

Linux ARM 32 bits

Linux ARM 64 bits

Mac OS X 10.10 or newer

[Release Notes](#)

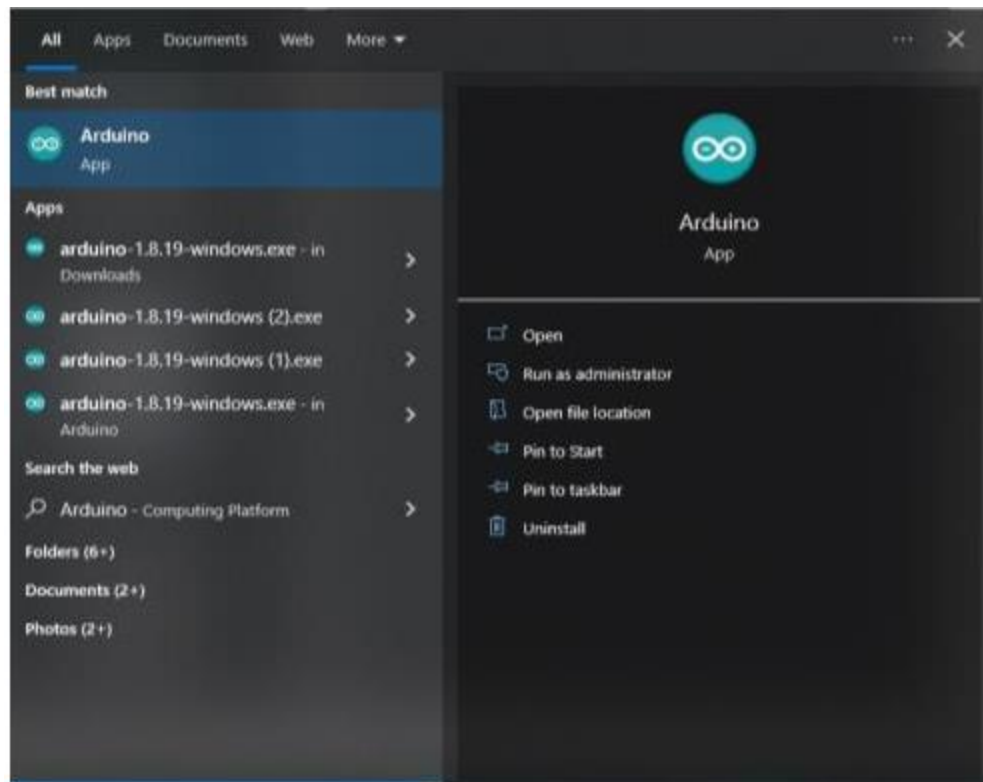
[Checksums \(sha512\)](#)

Step 3: Power up your board.

The Arduino Uno, Mega, Duemilanove and Arduino Nano automatically draw power from either, the USB connection to the computer or an external power supply. If you are using an Arduino Diecimila, you have to make sure that the board is configured to draw power from the USB connection. The power source is selected with a jumper, a small piece of plastic that fits onto two of the three pins between the USB and power jacks. Check that it is on the two pins closest to the USB port. Connect the Arduino board to your computer using the USB cable. The green power LED (labeled PWR) should glow.

Step 4: Launch Arduino IDE.

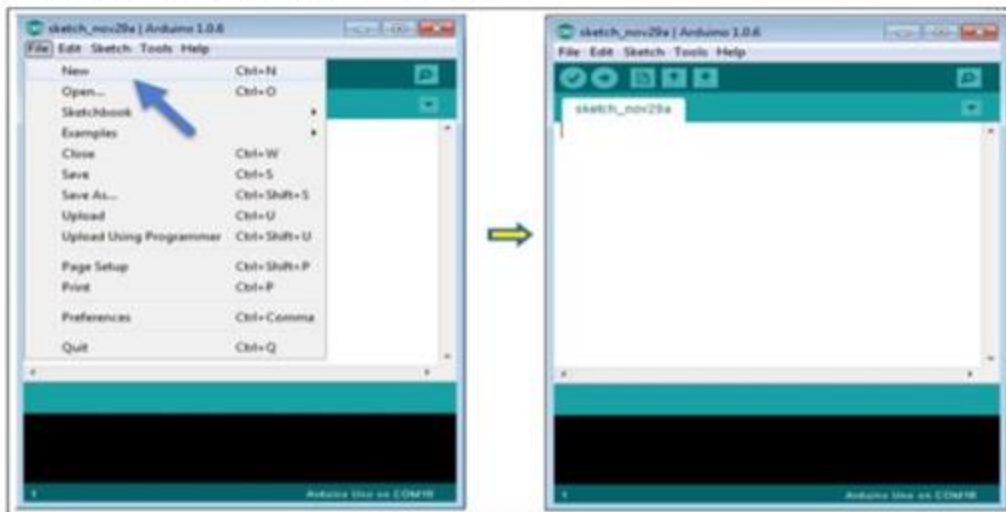
After your Arduino IDE software is downloaded, and Installed. Double-click the icon to start the IDE.



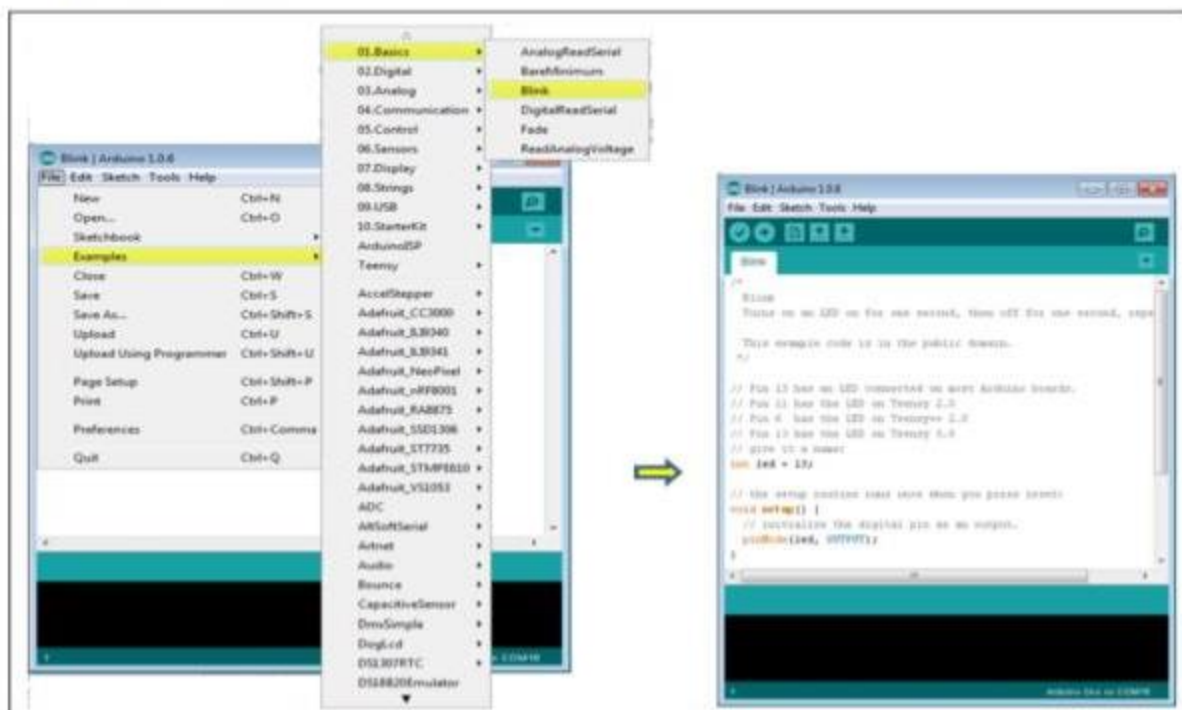
Step 5: Open your first project.

Once the software starts, you have two options
Create a new project.

To create a new project, select File --> New.



Open an existing project example.



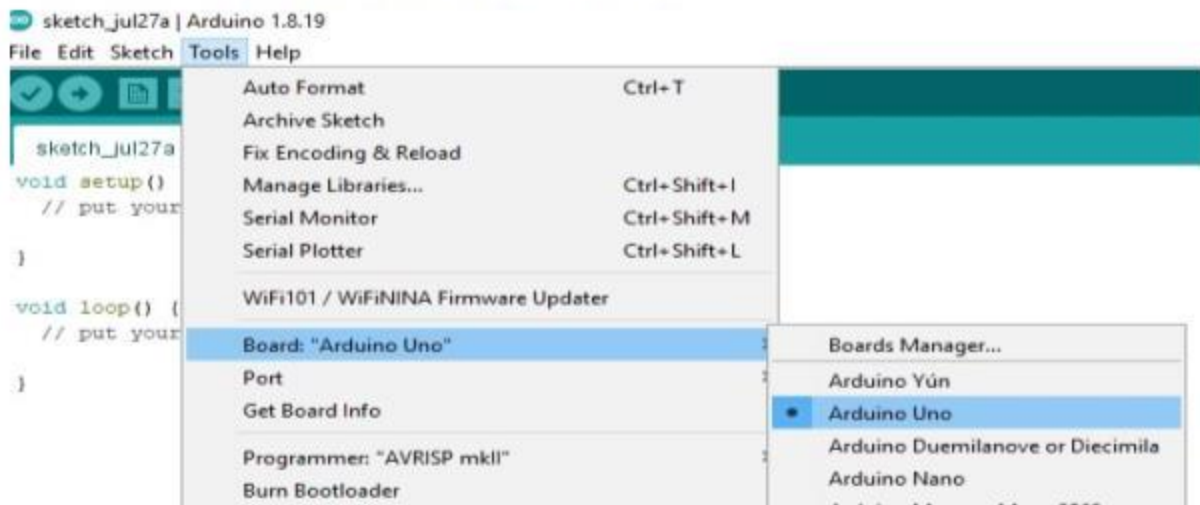
Here, we are selecting just one of the examples with the name Blink. It turns the LED on and off with some time delay. You can select any other example from the list.

Step 6: Select your Arduino board.

To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.

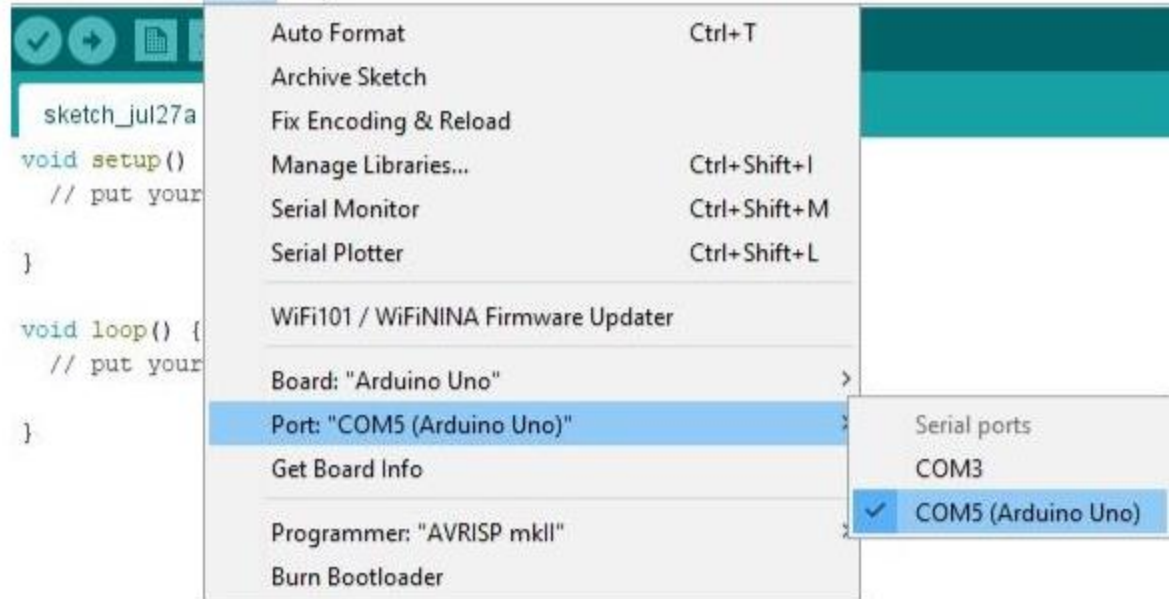
Go to Tools -> Board and select your board and Port

Go to Tools -> Board and select your board and Port



sketch_jul27a | Arduino 1.8.19

File Edit Sketch Tools Help



The screenshot shows the Arduino IDE interface. The 'Tools' menu is open, displaying various options. The 'Port: "COM5 (Arduino Uno)"' option is highlighted, which has opened a submenu. In this submenu, 'COM5 (Arduino Uno)' is selected with a checkmark. The background code editor shows parts of a C++ sketch for an Arduino Uno.

Tool	Shortcut
Auto Format	Ctrl+T
Archive Sketch	
Fix Encoding & Reload	
Manage Libraries...	Ctrl+Shift+I
Serial Monitor	Ctrl+Shift+M
Serial Plotter	Ctrl+Shift+L
WiFi101 / Wi-Fi NINA Firmware Updater	
Board: "Arduino Uno"	>
Port: "COM5 (Arduino Uno)"	>
Get Board Info	
Programmer: "AVRISP mkII"	>
Burn Bootloader	

Serial ports
COM3
✓ COM5 (Arduino Uno)

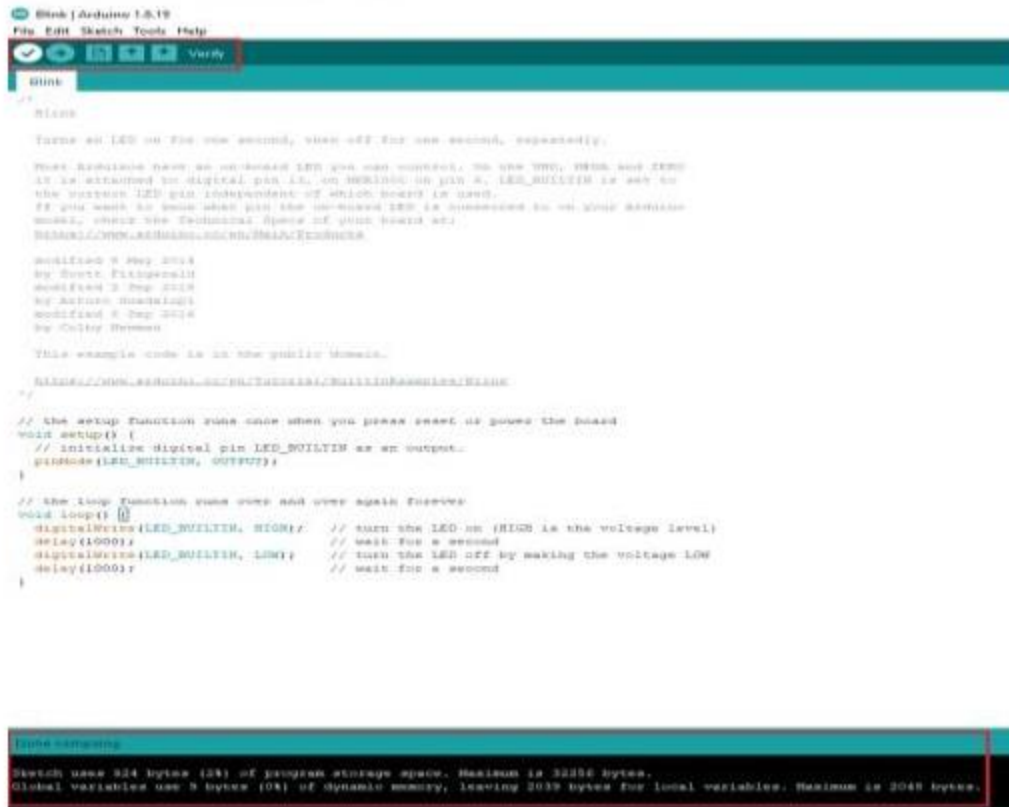

```
sketch_jul27a
void setup()
// put your code here

}

void loop() {
// put your code here
}
```

Step 7:

Compile and Upload the Code



The screenshot shows the Arduino IDE interface. At the top, the menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. Below the menu bar is a toolbar with icons for opening files, saving, and verifying code. The main text area contains the 'Blink' sketch code, which is a simple program that turns an LED on for one second and then off for one second, repeating this cycle. The code is written in C++ and includes comments explaining its functionality. At the bottom of the IDE, a status bar displays the following information: 'Sketch uses 324 bytes (3%) of program storage space. Maximum is 32256 bytes. Global variables use 0 bytes (0%) of dynamic memory, leaving 2048 bytes for local variables. Maximum is 2048 bytes.'

```
// Blink
// Turns an LED on for one second, then off for one second, repeatedly.

// Most Arduino have an on-board LED pin 13. On the Uno, this has BROWN
// It is attached to digital pin 13, on MEGA160 to pin 5. LED_BUILTIN is set to
// the correct LED pin independent of which board is used.
// If you want to know what pin the on-board LED is connected to on your Arduino
// board, check the Technical Specs of your board at:
// https://www.arduino.cc/en/Main/Products

// modified 8 May 2014
// by David Cuatrecasas
// modified 5 Sep 2014
// by Arturo Escobar
// modified 8 Sep 2014
// by Colby Brown

// This example code is in the public domain.

// https://www.arduino.cc/en/Tutorials/BuiltInExamples/Blink

//
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);                     // wait for a second
  digitalWrite(LED_BUILTIN, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);                     // wait for a second
}
```

Sketch uses 324 bytes (3%) of program storage space. Maximum is 32256 bytes.
Global variables use 0 bytes (0%) of dynamic memory, leaving 2048 bytes for local variables. Maximum is 2048 bytes.

Block | Arduino 1.8.19

File Edit Sketch Tools Help

Block

```
// =====  
  
// Turns an LED on for one second, then off for one second, repeatedly.  
  
// More details: there are two LEDs on the board; on the left, there are three  
// as is connected to digital pin 13, on the right on pin 4. LED_BUILTIN is set to  
// the correct LED pin independent of which board is used.  
// If you want to know what pin the onboard LED is connected to on your Arduino  
// board, check the Technical Specs of your board at:  
// https://www.arduino.cc/en/Hardware/Arduino  
  
// Modified 3 May 2014  
// by David Cuatrecasas  
// Modified 3 Sep 2014  
// by Andrew Burt  
// Modified 4 Sep 2014  
// by Collin Meehan  
  
// This example code is in the public domain.  
// https://www.arduino.cc/en/Tutorial/BuiltInExamples/Block  
//  
  
// The setup function runs once when you press reset or power the board  
void setup() {  
  // Initialize digital pin LED_BUILTIN as an output.  
  pinMode(LED_BUILTIN, OUTPUT);  
}  
  
// The loop function runs over and over again forever  
void loop() {  
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)  
  delay(1000); // wait for a second  
  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW  
  delay(1000); // wait for a second  
}
```

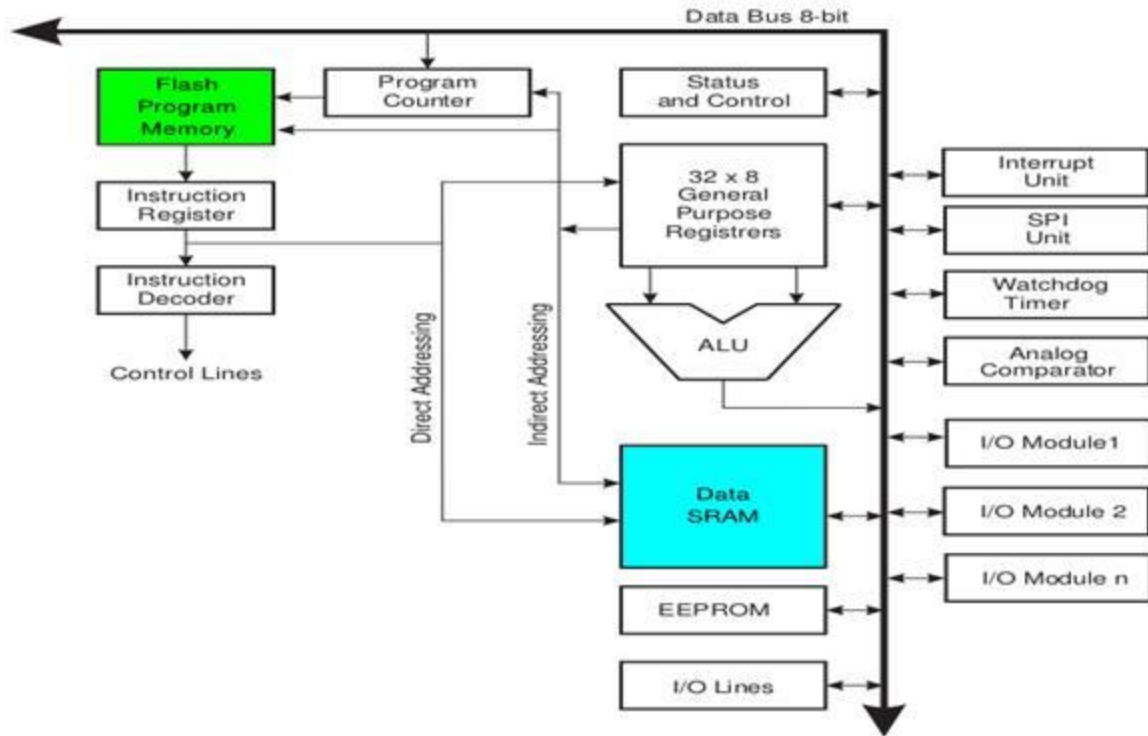
Block containing

Sketch uses 924 bytes (2%) of program storage space. Maximum is 32256 bytes.

Global variables use 0 bytes (0%) of dynamic memory, leaving 2048 bytes for local variables. Maximum is 2048 bytes.

Arduino Architecture:

- Arduino's processor basically uses the Harvard architecture where the program code and program data have separate memory.
- It consists of two memories- Program memory and the data memory.
- The code is stored in the flash program memory, whereas the data is stored in the data memory.
- The Atmega328 has 32 KB of flash memory for storing code (of which 0.5 KB is used for the bootloader), 2 KB of SRAM and 1 KB of EEPROM and operates with a clock speed of 16MHz.



Features of the Arduino Uno Board:

- It is an easy USB interface. This allows interface with USB as this is like a serial device.
- The chip on the board plugs straight into your USB port and supports on your computer as a virtual serial port.
- It is easy-to-find the microcontroller brain which is the ATmega328 chip. It has more number of hardware features like timers, external and internal interrupts, PWM pins and multiple sleep modes.
- It is an **open source design** and there is an advantage of being open source is that it has a large community of people using and troubleshooting it.
- It is a 16 MHz clock which is fast enough for most applications

- It is very convenient to manage power inside it and it had a feature of built-in voltage regulation.
- It has a 32 KB of flash memory for storing your code.
- 13 digital pins and 6 analog pins. This sort of pins allows you to connect hardware to your Arduino Uno board externally.

Concept of digital and analog ports,

- The Arduino can input and output analog signals as well as digital signals.
- An analog signal is one that can take on **any number of values**, unlike a digital signal which has only two values: **HIGH and LOW**.
- To measure the value of analog signals, the Arduino has a built-in analog-to-digital converter (ADC). The ADC turns the analog voltage into a digital value.
- The function that you use to obtain the value of an analog signal is `analogRead(pin)`.
- This function converts the value of the voltage on an analog input pin and returns a digital value from 0 to 1023, relative to the reference value.
- The default reference voltage is 5 V (for 5 V Arduino boards) or 3.3 V (for 3.3 V Arduino boards).
- It has one parameter which is the pin number.

- The Arduino does not have a built-in digital-to-analog converter (DAC), but it can **pulse-width modulate (PWM)** a digital signal to achieve some of the functions of an analog output.
- The function used to output a PWM signal is **`analogWrite(pin, value)`**.
- **pin** is the **pin number** used for the PWM output.
- **value** is a number proportional to the duty cycle of the signal. When **value = 0, the signal is always off**.
When **value = 255, the signal is always on**.
- On most Arduino boards, the PWM function is available on pins 3, 5, 6, 9, 10, and 11.
- The frequency of the PWM signal on most pins is approximately 490 Hz. On the Uno and similar boards, pins 5 and 6 have a frequency of approximately 980 Hz. Pins 3 and 11 on the Leonardo also run at 980 Hz.

- To map an analog input value, which ranges from 0 to 1023 to a PWM output signal, which ranges from 0 - 255, you can use the `map(value, fromLow, fromHigh, toLow, toHigh)` function.
- This function has five parameters, one is the variable in which the analog value is stored, while the others are 0, 1023, 0 and 255 respectively.

Types of Arduino board

- **Arduino Uno** is the most popular and widely used development board. It is powered by an **ATMega328P microcontroller**. It is the most popular choice among the community because it's, cheap, easy to learn and use, and also a variety of premade modules are available for this which makes it easier for developing new projects or prototypes.
- **Arduino Nano** is a small breadboard-friendly version of Arduino UNO. It has more or less functionality of the Arduino UNO but in a small form factor.
- **Arduino Pro mini** is actually a cut-down version of Nano. It has most of the functionalities similar to Arduino Nano but it lacks the onboard USB port.
- **Arduino Micro** is actually a Leonardo in a small form-factor **breadboard-friendly sized board**.
- **Arduino Mega 2560** is the biggest of all the boards. It is designed for applications where a lot of I/O or peripherals are needed. It is powered by a bigger and more capable processor the **ATMega2560**.

Thank you